Crimes in Philadelphia

Team 3

Data Science Capstone Project
Exploratory Data Analytics Report

Date:

11/15/2020

Team Members:

Name: Raj Patel

Name: Hong Son

Name: Kunal Sharma

[The purpose of this report is to describe the exploratory data analytics. It includes five major sections:

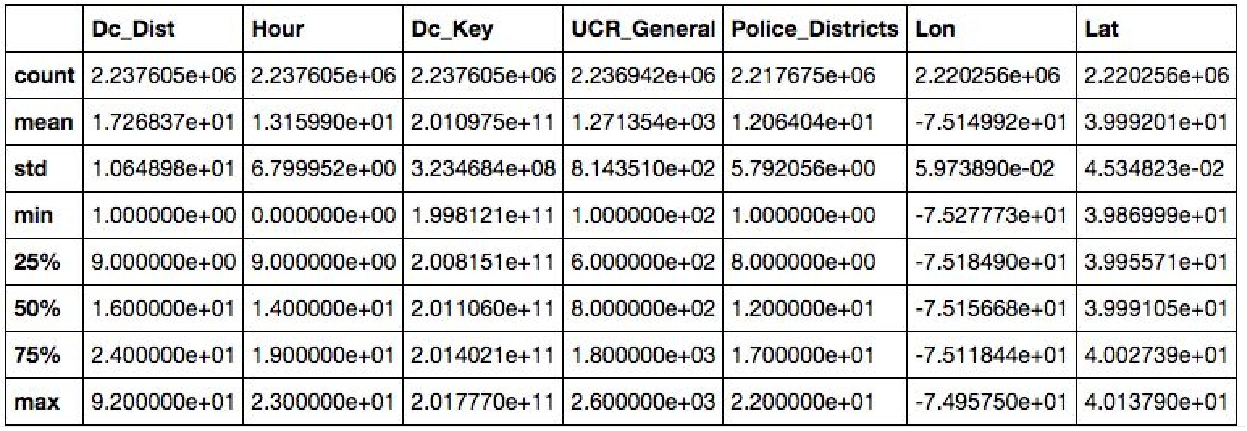
1. Analyzing the basic metrics of variables: data types, size, descriptive statistics
2. Non-graphical and graphical univariate analysis: identifying unique value and counts, histogram, box plots, etc.
3. Missing value analysis and outlier analysis
4. Feature engineering and analysis: correlation analysis, dimensionality reduction, deriving new variables
5. Appendix]

Analysis the basic metrics of variables

[In this section, we identify all the variables in the dataset and conduct the basic metrics of the variables. What are the data types (numerical/categorical, discrete or continuous, ordinal or nominal) and size? Provide the descriptive statistics of the variables such as mean, standard deviation, min, max, percentiles, etc.]

Philadelphia Crimes

Field Name	Alias	Description	Type
DC_Dist	District	A two character field that names the District boundary.	int64
DC_Key	DC Number	The unique identifier of the crime that consists of Year + District + Unique ID.	int64
Dispatch_Date_Time	Dispatch Date/Time	The date and time that the officer was dispatched to the scene.	object
Hour		The generalized hour of the dispatched time.	int64
Location_Block	Location Block	The location of crime generalized by street block.	object
Sector	PSA	A single character field that names the Police Service Area boundary.	object
Text_General_Code	General Crime Category	The generalized text for the crime code.	object
UCR_General	UCR Code	The rounded crime code, i.e. 614 to 600 (More info at https://metadata.phila.gov/#home/datasetdetails/5543868920583086178c4f8e/representationdetails/570e7621c03327dc14f4b68d/)	float64
Police_Districts	Police Districts	The police district number	float64
Lon	Longitude	the angular distance of a place east or west	float64
Lat	Latitude	the angular distance of a place north or south of the earth's equator	float64

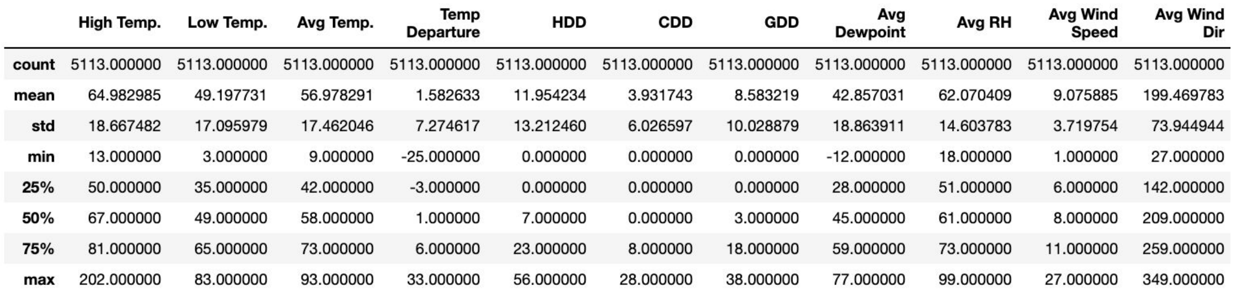


	Dc_Dist	Hour	Dc_Key	UCR_General	Police_Districts	Lon	Lat
count	2.237605e+06	2.237605e+06	2.237605e+06	2.236942e+06	2.217675e+06	2.220256e+06	2.220256e+06
mean	1.726837e+01	1.315990e+01	2.010975e+11	1.271354e+03	1.206404e+01	-7.514992e+01	3.999201e+01
std	1.064898e+01	6.799952e+00	3.234684e+08	8.143510e+02	5.792056e+00	5.973890e-02	4.534823e-02
min	1.000000e+00	0.000000e+00	1.998121e+11	1.000000e+02	1.000000e+00	-7.527773e+01	3.986999e+01
25%	9.000000e+00	9.000000e+00	2.008151e+11	6.000000e+02	8.000000e+00	-7.518490e+01	3.995571e+01
50%	1.600000e+01	1.400000e+01	2.011060e+11	8.000000e+02	1.200000e+01	-7.515668e+01	3.999105e+01
75%	2.400000e+01	1.900000e+01	2.014021e+11	1.800000e+03	1.700000e+01	-7.511844e+01	4.002739e+01
max	9.200000e+01	2.300000e+01	2.017770e+11	2.600000e+03	2.200000e+01	-7.495750e+01	4.013790e+01

Weather

Field Name	**Alias**	**Description**	**Type**
Date		The date of the collection	object
High Temp.	High Temperature	The highest temperature recorded that day	float64
Low Temp.	Low Temperature	The lowest temperature recorded that day	float64
Avg Temp.	Average Temperature	The Average temperature of the temperatures recorded	float64
Temp Departure			float64
HDD	heating degree day	A heating degree day (HDD) is a measurement designed to quantify the demand for energy needed to heat a building.	float64
CDD	Cooling degree day	A cooling degree day (CDD) is a measurement designed to quantify the demand for energy needed to cool buildings.	float64
GDD	growing degree days	Corn growing degree days (GDD) are calculated by subtracting the plant's lower base or threshold temperature of 50 °F (10 °C) from the average daily air temperature in °F or °C.	float64
Avg Dewpoint	Average Dewpoint	The dew point is the temperature to which air must be cooled to become saturated with water vapor.	float64
Avg RH	Relative humidity	Relative humidity (RH) is the ratio of the partial pressure of water vapor to the equilibrium vapor pressure of water at a given temperature.	float64
Avg Wind Speed	Average Wind	The Average Wind Speed of the wind recorded	float 64

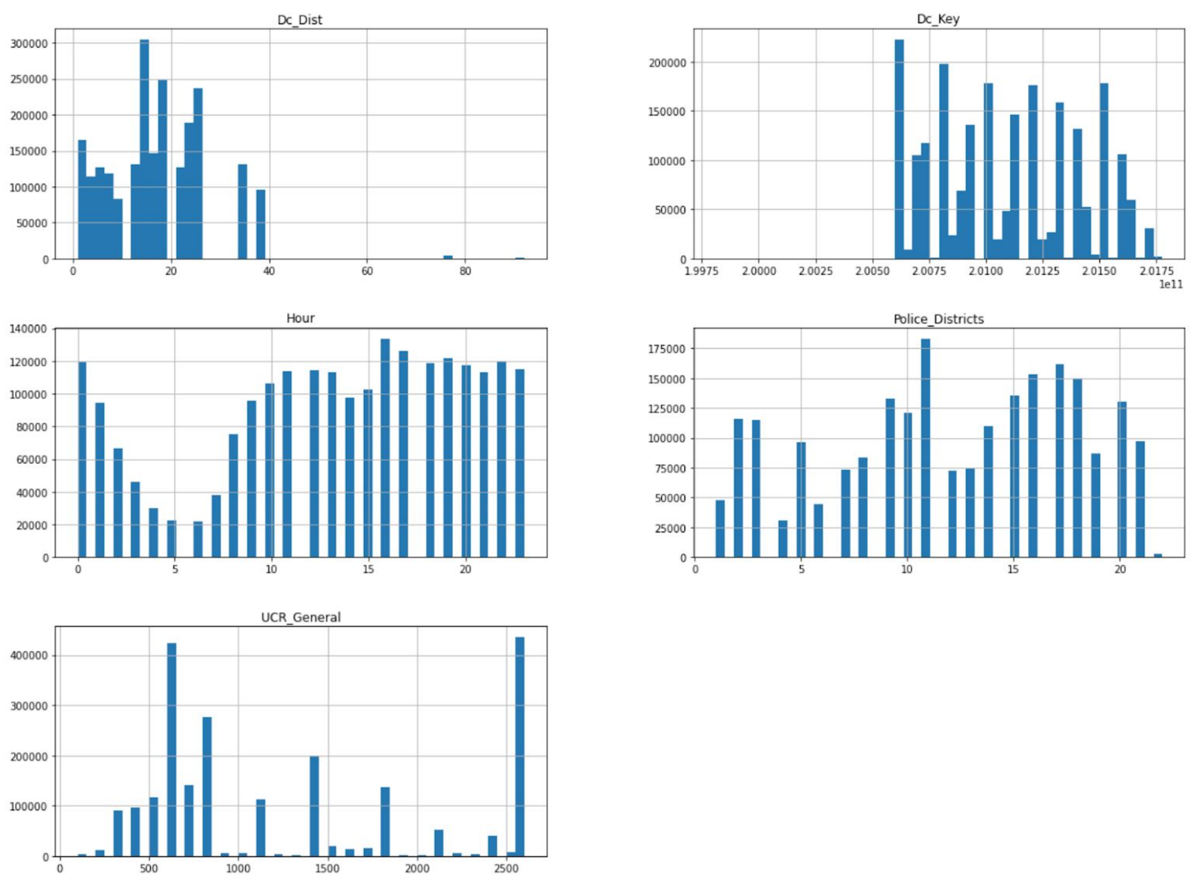
	Speed		
Avg Wind Dir	Average Wind Direction	The Average Wind Direction of the wind recorded	float64
Avg Press		Atmospheric pressure, also known as barometric pressure, is the pressure within the atmosphere of Earth	float64
Total Precip		NOTE: "Trace" amounts are defined as less than half that amount (0.005 inch).	object
Num Observations		The total number of observations for that day	float64



	High Temp.	Low Temp.	Avg Temp.	Temp Departure	HDD	CDD	GDD	Avg Dewpoint	Avg RH	Avg Wind Speed	Avg Wind Dir
count	5113.000000	5113.000000	5113.000000	5113.000000	5113.000000	5113.000000	5113.000000	5113.000000	5113.000000	5113.000000	5113.000000
mean	64.982985	49.197731	56.978291	1.582633	11.954234	3.931743	8.583219	42.857031	62.070409	9.075885	199.469783
std	18.667482	17.095979	17.462046	7.274617	13.212460	6.026597	10.028879	18.863911	14.603783	3.719754	73.944944
min	13.000000	3.000000	9.000000	-25.000000	0.000000	0.000000	0.000000	-12.000000	18.000000	1.000000	27.000000
25%	50.000000	35.000000	42.000000	-3.000000	0.000000	0.000000	0.000000	28.000000	51.000000	6.000000	142.000000
50%	67.000000	49.000000	58.000000	1.000000	7.000000	0.000000	3.000000	45.000000	61.000000	8.000000	209.000000
75%	81.000000	65.000000	73.000000	6.000000	23.000000	8.000000	18.000000	59.000000	73.000000	11.000000	259.000000
max	202.000000	83.000000	93.000000	33.000000	56.000000	28.000000	38.000000	77.000000	99.000000	27.000000	349.000000

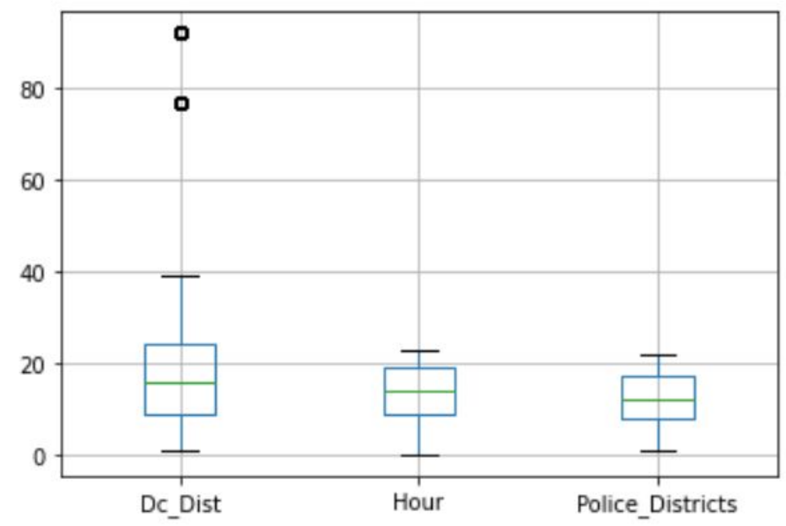
Non-graphical and graphical univariate analysis

[In this section, we identify the list and number of unique values for each variable and provide the histogram and box plots to understand the distribution of the data.]



```
crimeData.boxplot(column=['Dc_Dist', 'Hour', 'Police_Districts'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11aa33d90>
```



Missing value analysis and outlier analysis

[In this section, we identify the missing values and outliers and determine how we handle these values before analysis.]

We found missing values for the following columns:

```
#Printing name of each columns in the data set and number of na values

for crime in crimeData:
    value = crimeData[crime].isnull().sum()
    if (value>0):
        print("There are", value, " missing values in column",crime)
```

```
There are 663  missing values in column UCR_General
There are 663  missing values in column Text_General_Code
There are 19930  missing values in column Police_Districts
There are 17349  missing values in column Lon
There are 17349  missing values in column Lat
```

After reviewing the data, we determined to remove the rows with values with nan on geo location [lat, lon].

```
#Cleaning the data by deleting rows with values with nan on geo location

crimeData.replace('', float('NaN'), inplace = True)
crimeData.dropna(subset = ["Lon"], inplace=True)
crimeData.dropna(subset = ["Lat"], inplace=True)
```

For missing values on the columns, we mainly just removed the rows with missing values [1]. They make up a small portion of the entire dataset (<1%).

For our weather data, we were missing values for total precipitation but that made sense as missing values indicated no precipitation. Values of TRACE were converted to the value 0.001 to indicate that there was very little precipitation but not that there wasn't any.

Feature engineering and analysis

[In this section, we identify the variables that are useful for predictive modeling and machine learning through correlation analysis. You may also reduce the dimension or derive new variables so that the predictive modeling can be more efficient and effective.]

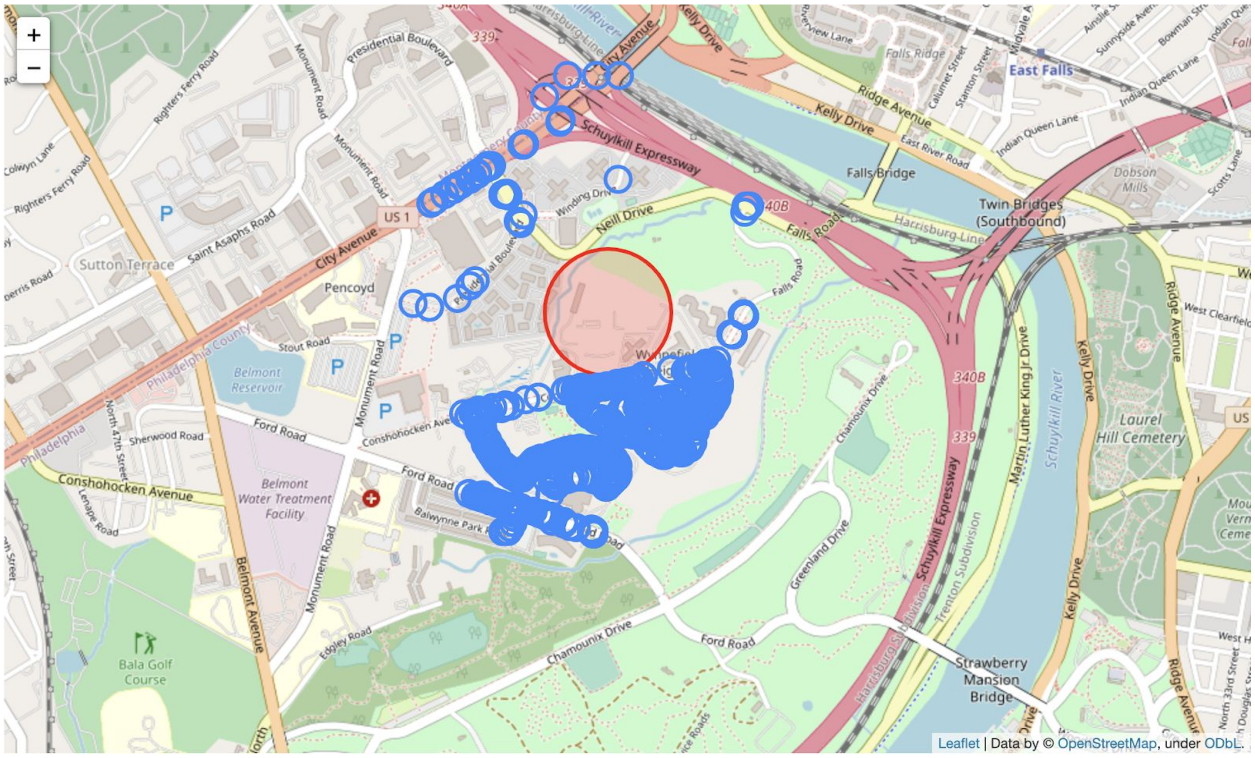
To gain a probability that a crime might occur given a user specified location and day of the year we have performed the following.

We are using Google's cloud API to gain the geolocations of any user specified location within Philadelphia in real time.

```
location = input("Enter a location in philly: \n")
geo = gmaps.geocode(location)
lat = geo[0]['geometry']['location']['lat']
lon = geo[0]['geometry']['location']['lng']
```

Philadelphia has more than 500,000 different points of geo-location and therefore once we gained the geo-location above we are also considering 500 nearest locations from the crime data set. To gain the nearest locations we are using k nearest neighbor from sklearn. [4]

```
from sklearn.neighbors import NearestNeighbors
knn = NearestNeighbors(n_neighbors=500)
knn.fit(X)
```



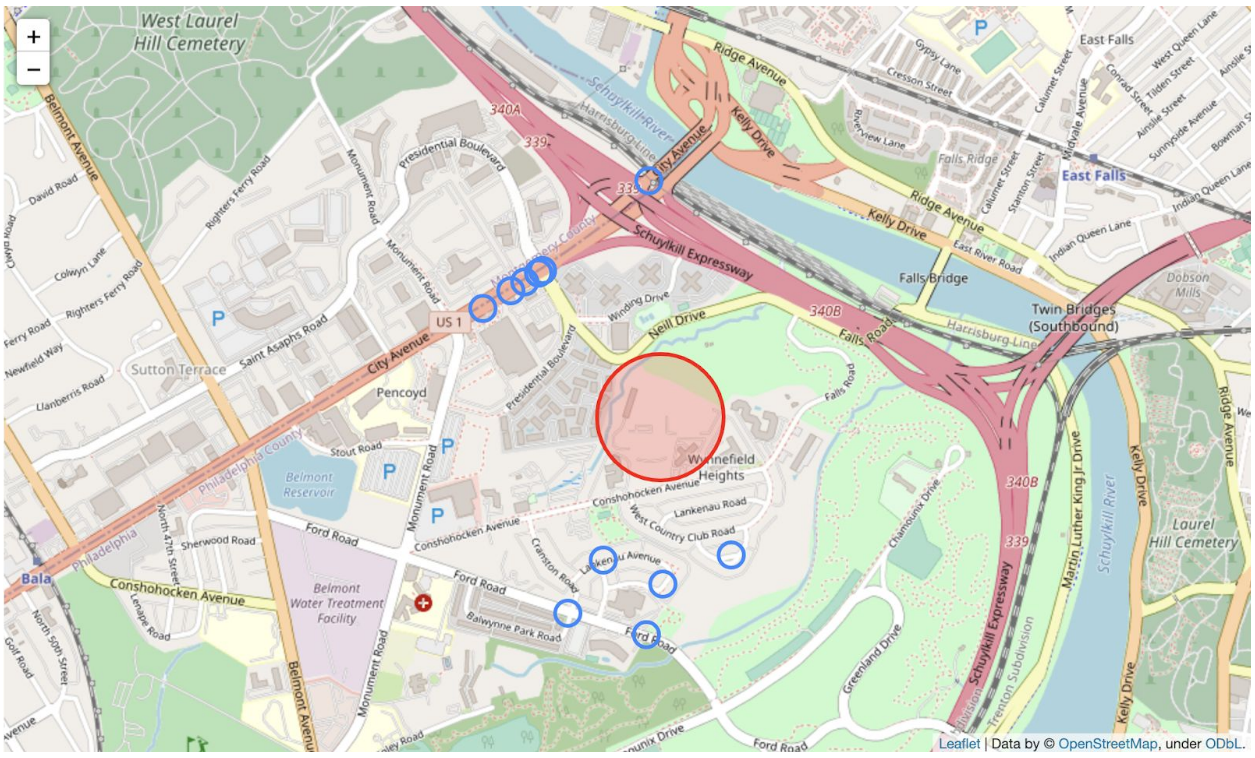
The above image contains the location that we focused on with a red marker and the nearest 500 different locations where crimes have occured in the past years from our dataset.

Since we want to focus only on the user specified day of the year, we filtered our dataset to have only the day (eg: Nov 11th) for each year in the dataset that we are using and the nearest locations from above.

```python
#Filtering the dataframe to to only have data from given day and month

nearestLocationsCrimesDF = nearestLocationsCrimesDF[nearestLocationsCrimesDF.Dispatch_Date.isin(dates)].copy()
nearestLocationsCrimesDF
```

	Dc_Dist	Psa	Dispatch_Date_Time	Dispatch_Date	Dispatch_Time	Hour	Dc_Key	Location_Block	UCR_General	Text_General_Code	...	Avg Wind Speed
36938	19	3	2009-11-13 00:27:00	2009-11-13	00:27:00	0	200919102491	3900 BLOCK CITY AV	1400.0	Vandalism/Criminal Mischief	...	22
166514	19	3	2010-11-13 23:31:00	2010-11-13	23:31:00	23	201019106480	4000 BLOCK FORD RD	600.0	Theft from Vehicle	...	4
346244	19	3	2011-11-13 16:35:00	2011-11-13	16:35:00	16	201119101011	4000 BLOCK CITY AVE	600.0	Thefts	...	10
355068	19	3	2011-11-13 14:16:00	2011-11-13	14:16:00	14	201119100989	3900 BLOCK LANKENAU AVE	2600.0	All Other Offenses	...	10
374907	19	3	2011-11-13 21:12:00	2011-11-13	21:12:00	21	201119101097	3900 BLOCK CITY AVE	500.0	Burglary Residential	...	10
401237	19	3	2015-11-13 21:47:00	2015-11-13	21:47:00	21	201519115744	CITY AV / SCHUYLKILL EXPY RAMP A	2100.0	DRIVING UNDER THE INFLUENCE	...	16
448235	19	3	2015-11-13 09:56:00	2015-11-13	09:56:00	9	201519115544	4000 BLOCK CITY AVE	300.0	Robbery No Firearm	...	16
933183	19	3	2013-11-13 22:37:00	2013-11-13	22:37:00	22	201319099277	3900 BLOCK FORD RD	800.0	Other Assaults	...	10
1006982	19	3	2014-11-13 14:39:00	2014-11-13	14:39:00	14	201419104712	4000 BLOCK CITY AVE	600.0	Thefts	...	8
1016674	19	3	2014-11-13 00:25:00	2014-11-13	00:25:00	0	201419104509	3700 BLOCK MIMI CIR	1100.0	Fraud	...	8



The image above contains only the nearest locations where the crimes have occurred on a user specified date (Nov 11th in this case). In the filtered data frame above based on user specified day and location, we have also achieved essential data about the weather, wind speed, type of crime (violent or non-violent).

Finally, to gain a probability that a crime will occur on a user specified data and location we get the total number of years when a crime has occured in our dataset and divide it with the total number of years from the dataset.

```
topCrimeCasesNearestLocation["event"] = 1
probabilityOfCrime = topCrimeCasesNearestLocation["event"].sum() / len(dates)
probabilityOfCrime
```

```
0.5833333333333334
```

Appendix

[Provide the code or pseudo code, and any other information in the appendix here.]

1.

```
In [1]: #Importing Libraries

        import pandas as pd
        import numpy as np
        from matplotlib import pyplot as plt
        import seaborn as sns
        import os
        from scipy.stats import f_oneway
        import plotly.express as px
        import datetime
        import googlemaps
        import folium
```

```
In [2]: #Reading Data from a csv file

        crimeData = pd.read_csv("crime.csv")
        crimeData.sample(5)
```

Out[2]:

	Dc_Dist	Psa	Dispatch_Date_Time	Dispatch_Date	Dispatch_Time	Hour	Dc_Key	Location_Block	UCR_General	Text_General_Code	Police_Dist
131181	7	3	2010-09-30 01:46:00	2010-09-30	01:46:00	1	201007037482	13000 BLOCK KELVIN AVE	1800.0	Narcotic / Drug Law Violations	
2066557	25	3	2016-05-10 09:33:00	2016-05-10	09:33:00	9	201625038389	100 BLOCK W CAMBRIA ST	2600.0	All Other Offenses	
1497806	25	N	2007-07-13 04:35:00	2007-07-13	04:35:00	4	200725074900	1200 BLOCK W CAMBRIA ST	500.0	Burglary Residential	
321887	3	3	2011-11-18 10:12:00	2011-11-18	10:12:00	10	201103077698	1000 BLOCK MC KEAN ST	500.0	Burglary Residential	
585509	8	2	2012-06-05 10:23:00	2012-06-05	10:23:00	10	201208023791	9100 BLOCK ACADEMY RD	1000.0	Forgery and Counterfeiting	

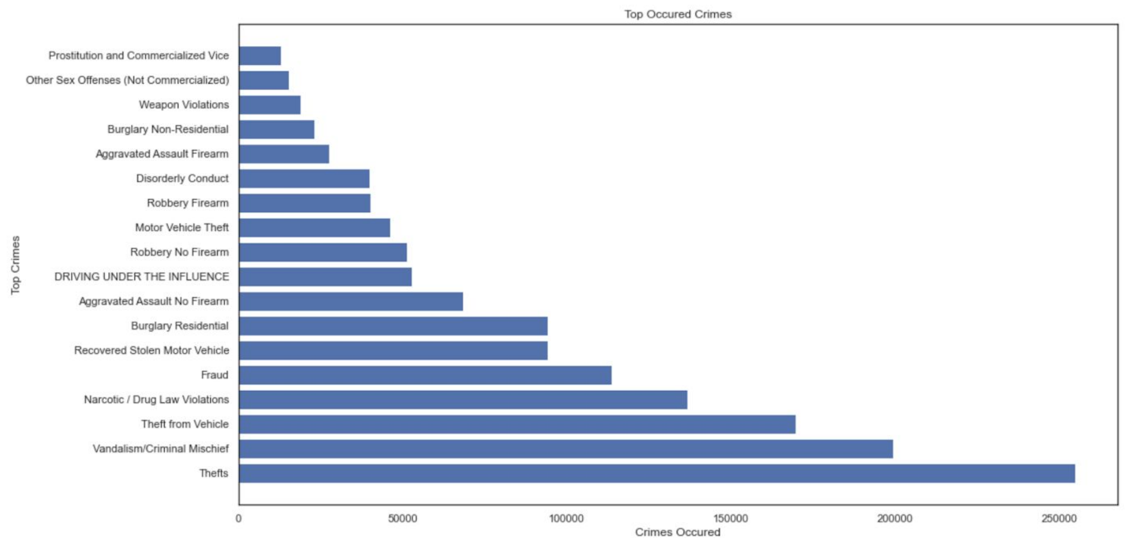
```
In [3]: #Cleaning the data by deleting rows with values with nan on geo location

        crimeData.replace('', float('NaN'), inplace = True)
        crimeData.dropna(subset = ["Lon"], inplace=True)
        crimeData.dropna(subset = ["Lat"], inplace=True)
```

```
#Plotting a graph to visualize each crime occured each day over last 10 years

fig, ax = plt.subplots(figsize=(16, 9))
ax.barh(topCrimesList, topCrimeData)
ax.set(xlabel='Crimes Occured', ylabel='Top Crimes',
       title='Top Occured Crimes')
```

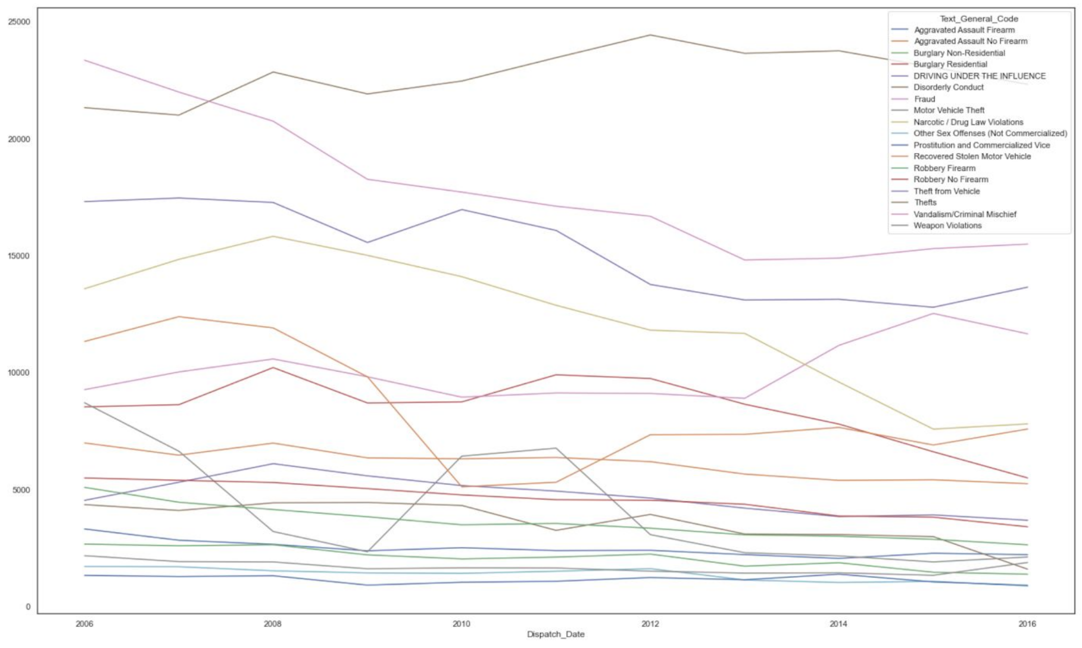
```
)ut[370]: [Text(0, 0.5, 'Top Crimes'),
           Text(0.5, 0, 'Crimes Occured'),
           Text(0.5, 1.0, 'Top Occured Crimes')]
```



2.

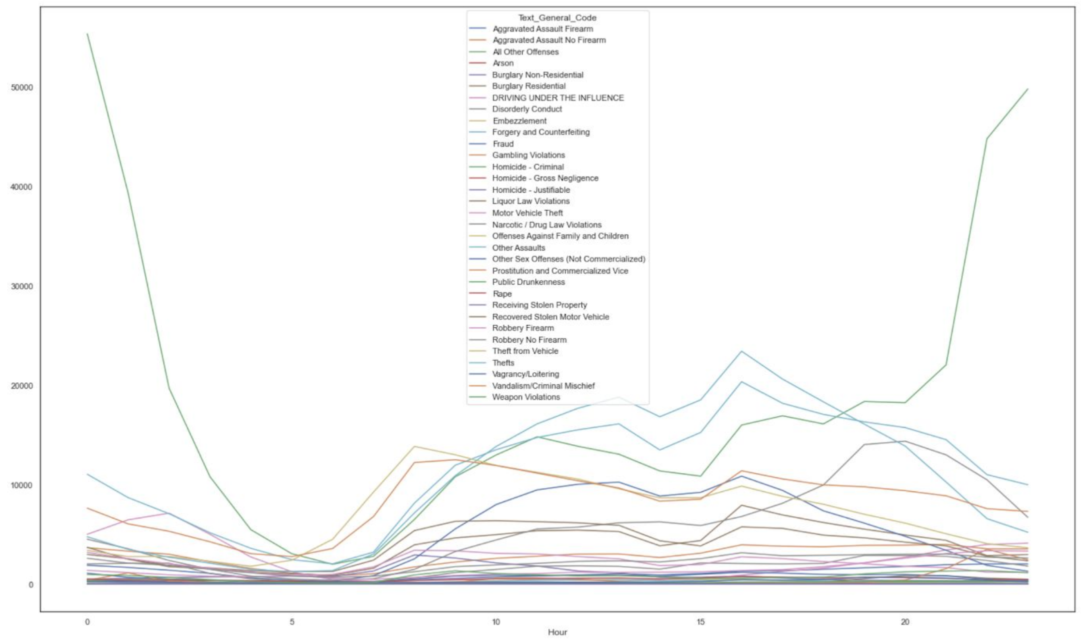
```
topCrimeCasesByYear.plot()
```

```
Out[378]: <matplotlib.axes._subplots.AxesSubplot at 0x7feb40eb4f70>
```



```
crimeDataByHour.plot()
```

Out[384]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb3f56f490>
```



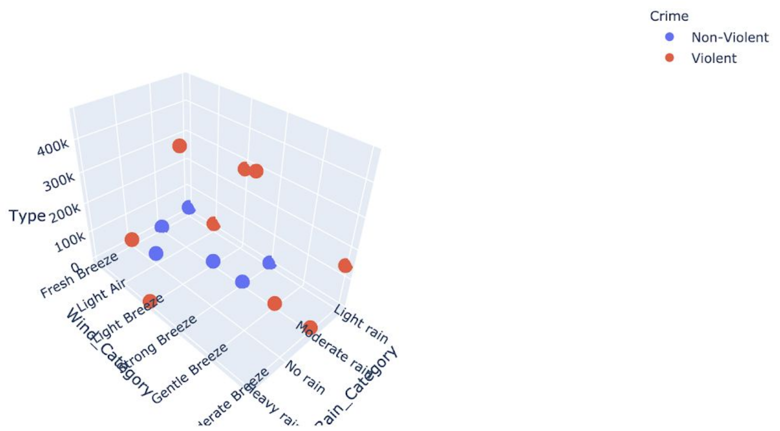
In [376]:
```
#Grouping by year to get total number of each top occured crime for further analysis

topCrimeCasesByYear = topCrimeCasesByYear.groupby('Dispatch_Date').sum()
topCrimeCasesByYear
```

Out[376]:

Text_General_Code	Aggravated Assault Firearm	Aggravated Assault No Firearm	Burglary Non-Residential	Burglary Residential	DRIVING UNDER THE INFLUENCE	Disorderly Conduct	Fraud	Motor Vehicle Theft	Narcotic / Drug Law Violations	Other Sex Offenses (Not Commercialized)	Prostitution and Commercialized Vice	R
Dispatch_Date												
2006	3312.0	6990.0	2665.0	8533.0	4536.0	4354.0	9269.0	8709.0	13583.0	1708.0	1327.0	
2007	2830.0	6468.0	2591.0	8626.0	5309.0	4103.0	10026.0	6636.0	14840.0	1697.0	1277.0	
2008	2656.0	6984.0	2637.0	10213.0	6103.0	4429.0	10582.0	3203.0	15827.0	1523.0	1311.0	
2009	2378.0	6352.0	2204.0	8699.0	5583.0	4445.0	9820.0	2338.0	15012.0	1433.0	911.0	
2010	2511.0	6312.0	2032.0	8745.0	5169.0	4318.0	8948.0	6426.0	14100.0	1413.0	1033.0	
2011	2388.0	6369.0	2109.0	9905.0	4931.0	3252.0	9129.0	6767.0	12876.0	1504.0	1076.0	
2012	2400.0	6190.0	2237.0	9746.0	4632.0	3934.0	9107.0	3069.0	11813.0	1615.0	1235.0	
2013	2216.0	5659.0	1722.0	8646.0	4202.0	3095.0	8899.0	2297.0	11674.0	1132.0	1142.0	
2014	2052.0	5388.0	1868.0	7797.0	3842.0	3073.0	11165.0	2160.0	9592.0	1024.0	1377.0	
2015	2274.0	5414.0	1461.0	6613.0	3914.0	2985.0	12529.0	1902.0	7582.0	1072.0	1049.0	
2016	2214.0	5248.0	1377.0	5494.0	3685.0	1593.0	11656.0	2112.0	7805.0	875.0	903.0	

```python
import plotly.express as px
temp_grp_df = weather_crime_data.groupby(['Wind_Category', 'Rain_Category']).count().reset_index(drop=False)
temp_grp_df = temp_grp_df.assign(Crime=weather_crime_data['Type'])
temp_grp_df = temp_grp_df.dropna()
fig = px.scatter_3d(temp_grp_df, y='Wind_Category', x='Rain_Category', z="Type", color='Crime')
fig.show()
```

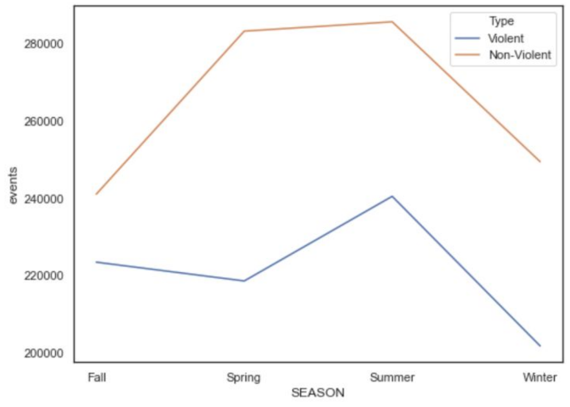


3.

```
In [82]: import seaborn as sns

         # flattened2 = pd.DataFrame(seasoncases.to_records()).sort_index(ascending=False)
         plt.figure(figsize=(8,6))
         sns.lineplot(data=flattened, x="SEASON", y="events", hue="Type")

Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x7fec9743ae20>
```



```
In [108]: # adding geo loaction in a list to later filter the weather_crime data frame

          folium_map = folium.Map(location=[lat, lon],
                                  zoom_start=16,
                                 )
          folium.CircleMarker(
              location=[lat, lon],
              radius=50,
              color='red',
              fill=True,
              fill_color='red'
          ).add_to(folium_map)
          nearestNeighborLonLat = []
          for i in testArray[0]:
              nearestNeighborLonLat.append(str(groupCrimeLatLon.iloc[i][1])+", "+str(groupCrimeLatLon.iloc[i][0]))
              folium.CircleMarker(
                  location=[groupCrimeLatLon.iloc[i][0],
                            groupCrimeLatLon.iloc[i][1]]
              ).add_to(folium_map)
          nearestNeighborLonLat[:5]

Out[108]: ['-75.189337, 39.956621000000005',
           '-75.18932099999999, 39.95661',
           '-75.189436, 39.956979',
           '-75.189308, 39.956226',
           '-75.189199, 39.956976']
```

4.

```
In [121]: # adding geo loaction in a list to later filter the weather_crime data frame

          filtered_map = folium.Map(location=[lat, lon],
                                  zoom_start=16)
          folium.CircleMarker(
              location=[lat, lon],
              radius=50,
              color='red',
              fill=True,
              fill_color='red'
          ).add_to(filtered_map)
          for i in range(nearestLocationsCrimesDF.shape[0]):
              folium.CircleMarker(
                  location=[nearestLocationsCrimesDF["Lat"].to_list()[i],
                            nearestLocationsCrimesDF["Lon"].to_list()[i]]
              ).add_to(filtered_map)
          filtered_map

Out[121]:
```

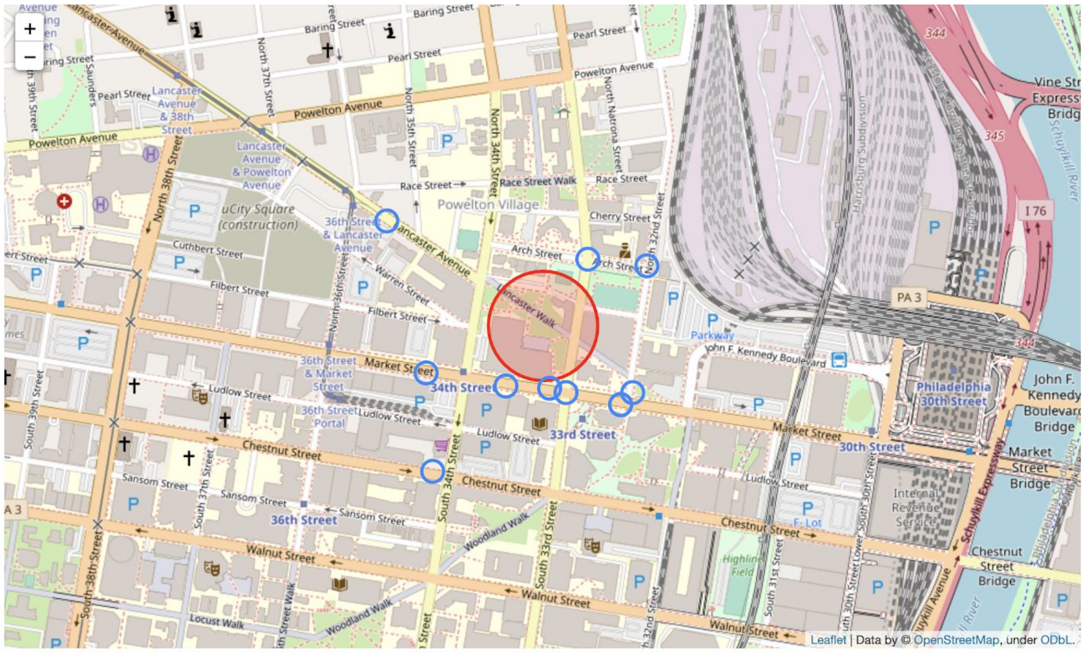


Table of Contributions

The table below identifies contributors to various sections of this document.

	Section	Writing	Editing
1	**Analysis the basic metrics of variables**	**Hong**	**Raj**
2	**Non-graphical and graphical univariate analysis**	**Hong**	**Raj**
3	**Missing value analysis and outlier analysis**	**Hong**	**Raj**
4	**Feature engineering and analysis**	**Kunal**	**Raj**
5	**Appendix**	**Kunal**	**Raj**

Grading

The grade is given on the basis of quality, clarity, presentation, completeness, and writing of each section in the report. This is the grade of the group. Individual grades will be assigned at the end of the term when peer reviews are collected.